Computer Science 330 Language Implementation Test

6.20-8.00pm Thursday 14th April 2005

Start reading 6.20p.m. Write your name on all sheets of your answer book. Start writing your answers at 6.30pm. Stop writing at 8.00p.m.

Remove the staple fastening the question sheets to the answer book, but do not remove the staples from the answer book. Read the questions carefully. Hand in your answer book at the front of the class. Always show your working - most marks are for showing you know what you are doing, rather than just getting the right answer. Attempt all questions. Questions total 100 marks. The test counts for 20% of the total mark.

Question 1 — 17 Marks

Write JFlex rules to match the following tokens.

(a) A binary integer, composed of the binary indicator "0b", followed by one or more binary digits. For example `0b101011`, but not `0b1234` or `0B101011` or `101011`.

(2 marks)

(b) A string composed of a double quote ("), zero or more component characters, then another double quote ("). A component character can be: any character except a control character (\0 to \037, \177) or backslash (\) or double quote ("); or can be a pair of double quotes ("", representing a single "). For example `"He said ""hello"""`, but not `"He said \"hello\""` or `"he said "hello""`.

(3 marks)

(c) A decimal integer, with ","s to separate the digits into groups of three, as is normally done when writing numbers in English. For example `12,345,678` but not `12345678` or `123,456,78`.

(7 marks)

(d) C style `/* ... */` comments, but allowing nesting of comments. For example,

```
/*
A comment.
/* A nested comment */
And more of the same comment.
*/
```

Add appropriate actions to increment the line count, when a line break occurs. Do not return a value.

(5 marks)

Question 2 63 marks

Consider the following CUP grammar.

```
terminal LEFTBRACE, RIGHTBRACE, CLASS, COMMA, SEMICOLON, ERROR;
terminal String IDENT;

non terminal ClassDecl;
non terminal Type;
non terminal DeclList;
non terminal Decl;
non terminal IdentList;

start with ClassDecl;

ClassDecl::=
            CLASS IDENT LEFTBRACE DeclList RIGHTBRACE
      ;

DeclList::=
            /* Empty */
      |
            Decl
      |
            Decl SEMICOLON DeclList
      ;

Decl::=
            Type IdentList
      |
            ClassDecl
      ;

Type::=
            IDENT
      ;

IdentList::=
            IDENT
      |
            IdentList COMMA IDENT
      ;
```

Assume that

- The terminal symbols LEFTBRACE, RIGHTBRACE, CLASS, COMMA, SEMICOLON correspond to "`{`", "`}`", "`class`", "`,`", "`;`".
- IDENT corresponds to an identifier.

(a) Using the information provided in the appendix, perform a shift-reduce LALR(1) parse of the input

```
class A { int a, b; A next; }
```

Show both the symbols and states on the stack, the current token, and the action performed at each stage. (Consider "int" to be an IDENT). (22 marks)

(b) Draw the full parse tree, showing all rules used in the above shift-reduce LALR(1) parse. (8 marks)

(c) (i) Note that DeclList is nullable.

(ii) Draw the first graph, and compute the first sets for this grammar.

(iii) Draw the follow graph, and compute the follow sets for this grammar. (20 marks)

(d) State 10 is

```
lalr_state [10]: {
  [Decl ::= (*) Type IdentList , {RIGHTBRACE SEMICOLON }]
  [DeclList ::= (*) , {RIGHTBRACE }]
  [DeclList ::= Decl SEMICOLON (*) DeclList , {RIGHTBRACE }]
  [Type ::= (*) IDENT , {IDENT }]
  [DeclList ::= (*) Decl SEMICOLON DeclList , {RIGHTBRACE }]
  [ClassDecl ::= (*) CLASS IDENT LEFTBRACE DeclList RIGHTBRACE ,
      {RIGHTBRACE SEMICOLON }]
  [Decl ::= (*) ClassDecl , {RIGHTBRACE SEMICOLON }]
  [DeclList ::= (*) Decl , {RIGHTBRACE }]
}
transition on Decl to state [9]
transition on IDENT to state [8]
transition on CLASS to state [2]
transition on ClassDecl to state [7]
transition on Type to state [6]
transition on DeclList to state [11]
```

(i) Write down the set of items for goto(state 10, Decl) (state 9).

(5 marks)

(ii) Write down the set of items for goto(state 10, Type) (state 6). Make sure you take the closure.

(8 marks)

Question 3 20 marks

Consider the structure of a textual description of a state generated by CUP, as in the example given in question 2(d).

For example, a state description starts with the keyword “lalr_state”, then an integer constant enclosed in “[...]”, then a “:”, then a list of items enclosed in “{...}”, then a list of transition specifications.

Write a grammar for the **general structure of a state description**.

Make it clear which symbols in your grammar for a state description are terminal symbols, and which are nonterminals. (Note that the symbols such as Decl, Type, etc, used in the example are all just IDENT, as far as your grammar for a state description is concerned.) For example, you could represent terminal symbols other than IDENT and INTCONST by enclosing them, in “...”.

Note that the righthand sides of rules in items have a single “(*)”, and do not have “|”s.

You do not have to write any actions.

Appendix

Tables for the CUP grammar

Grammar Rules (Productions)

```
9: IdentList ::= IdentList COMMA IDENT
8: IdentList ::= IDENT
7: Type ::= IDENT
6: Decl ::= ClassDecl
5: Decl ::= Type IdentList
4: DeclList ::= Decl SEMICOLON DeclList
3: DeclList ::= Decl
2: DeclList ::=
1: ClassDecl ::= CLASS IDENT LEFTBRACE DeclList RIGHTBRACE
0: $START ::= ClassDecl EOF
```

Action Table

```
From state #0
      CLASS:SHIFT(2)
From state #1
      EOF:SHIFT(17)
From state #2
      IDENT:SHIFT(3)
From state #3
      LEFTBRACE:SHIFT(4)
From state #4
      RIGHTBRACE:REDUCE(2) CLASS:SHIFT(2) IDENT:SHIFT(8)
From state #5
      RIGHTBRACE:SHIFT(16)
From state #6
      IDENT:SHIFT(12)
From state #7
      RIGHTBRACE:REDUCE(6) SEMICOLON:REDUCE(6)
From state #8
      IDENT:REDUCE(7)
From state #9
      RIGHTBRACE:REDUCE(3) SEMICOLON:SHIFT(10)
From state #10
      RIGHTBRACE:REDUCE(2) CLASS:SHIFT(2) IDENT:SHIFT(8)
From state #11
      RIGHTBRACE:REDUCE(4)
From state #12
      RIGHTBRACE:REDUCE(8) COMMA:REDUCE(8) SEMICOLON:REDUCE(8)
From state #13
      RIGHTBRACE:REDUCE(5) COMMA:SHIFT(14) SEMICOLON:REDUCE(5)
From state #14
      IDENT:SHIFT(15)
From state #15
      RIGHTBRACE:REDUCE(9) COMMA:REDUCE(9) SEMICOLON:REDUCE(9)
From state #16
      EOF:REDUCE(1) RIGHTBRACE:REDUCE(1) SEMICOLON:REDUCE(1)
From state #17
      EOF:REDUCE(0)
```

Reduce (GoTo) Table

```
From state #0:
      ClassDecl:GOTO(1)
From state #1:
From state #2:
From state #3:
From state #4:
      ClassDecl:GOTO(7)
      Type:GOTO(6)
      DeclList:GOTO(5)
      Decl:GOTO(9)
From state #5:
From state #6:
      IdentList:GOTO(13)
From state #7:
From state #8:
From state #9:
From state #10:
      ClassDecl:GOTO(7)
      Type:GOTO(6)
      DeclList:GOTO(11)
      Decl:GOTO(9)
From state #11:
From state #12:
From state #13:
From state #14:
From state #15:
From state #16:
From state #17:
```

This page left blank for formatting purposes

Computer Science 330

Language Implementation Test

Thursday 14th April 2005

Answer Booklet

Surname
Given Names
Student ID Number
Login Name
Normal Signature

1 _____/17

2(a) _____/22

2(b) _____/8

2(c) _____/20

2(d) _____/13

3 _____/20

Total _____/100

Do not write on this page.

It will not be returned to you.

Print your login name _____

Question 1 17 Marks

Write JFlex rules to match the following tokens, and where appropriate return a value, or perform some appropriate action.

Write JFlex rules to match the following tokens.

(a) A binary integer, composed of the binary indicator "0b", followed by one or more binary digits.

(2 marks)

(b) A string composed of a double quote ("), zero or more component characters, then another double quote ("). A component character can be: any character except a control character (\0 to \037, \177) or backslash (\) or double quote ("); or can be a pair of double quotes ("", representing a single ").

(3 marks)

(c) A decimal integer, with ","s to separate the digits into groups of three, as is normally done when writing numbers in English.

(7 marks)

(d) C style `/* ... */` comments, but allowing nesting of comments.

(5 marks)

Question 2 **63 marks**

(a) Using the information provided in the appendix, perform a shift-reduce LALR(1) parse of the input

```
class A { int a, b; A next; }
```

Show both the symbols and states on the stack, the current token, and the action performed at each stage. (22 marks)

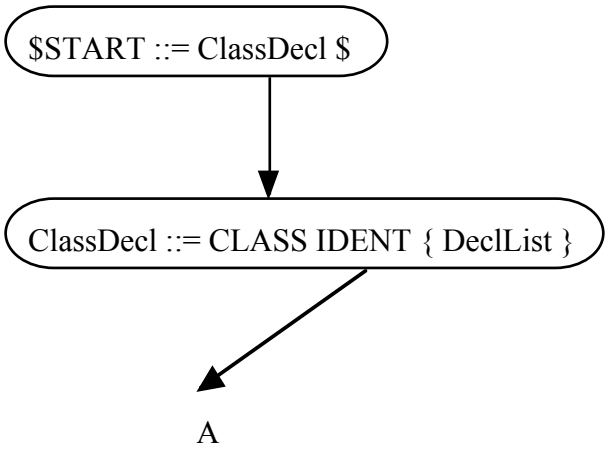
Stack								
$0								
$0	class 2							
$0	class 2	ID 3						
$0								
$0								
$0								
$0								
$0								
$0								
$0								
$0								
$0								
$0								
$0								
$0								
$0								
$0								
$0								
$0								
$0								
$0								
$0								
$0								
$0	CDec 1							
$0	CDec 1	$17						
$0	$Start -1							

Input	Action	
class	Shift	class 2
ID A	Shift	ID 3
{	Shift	{ 4
	Shift	$17
	Reduce	$start ::= ClassDecl $
	Accept	

Print your login name _____

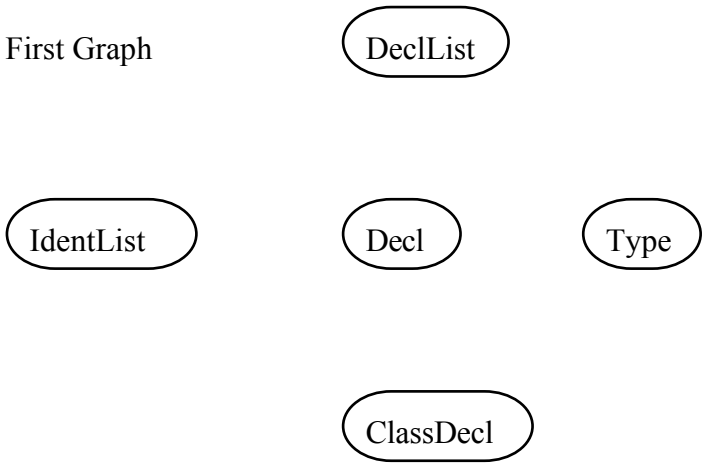
(b) Draw the full parse tree, showing all rules used in the above shift-reduce LALR(1) parse.

(8 marks)

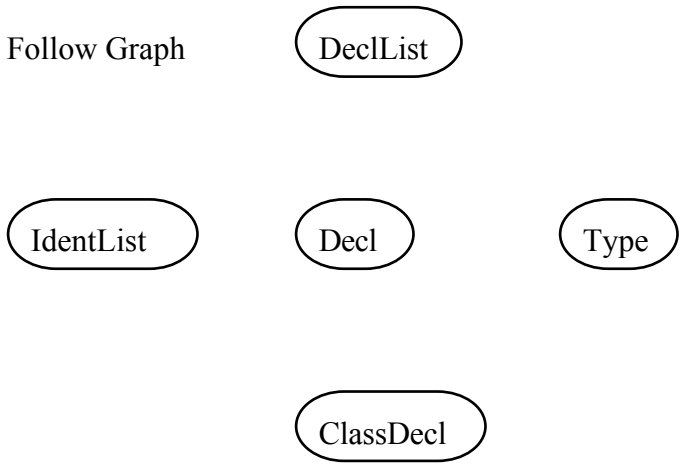


(c) (20 marks)

Draw the first graph for this grammar.



Draw the follow graph for this grammar.



Indicate the first and follow sets for the grammar

Symbol	First Set	Follow Set
DeclList		
Decl		
ClassDecl		
Type		
IdentList		

Print your login name _____

(d) State 10 is

```
lalr_state [10]: {
  [Decl ::= (*) Type IdentList , {RIGHTBRACE SEMICOLON }]
  [DeclList ::= (*) , {RIGHTBRACE }]
  [DeclList ::= Decl SEMICOLON (*) DeclList , {RIGHTBRACE }]
  [Type ::= (*) IDENT , {IDENT }]
  [DeclList ::= (*) Decl SEMICOLON DeclList , {RIGHTBRACE }]
  [ClassDecl ::= (*) CLASS IDENT LEFTBRACE DeclList RIGHTBRACE ,
      {RIGHTBRACE SEMICOLON }]
  [Decl ::= (*) ClassDecl , {RIGHTBRACE SEMICOLON }]
  [DeclList ::= (*) Decl , {RIGHTBRACE }]
}
transition on Decl to state [9]
transition on IDENT to state [8]
transition on CLASS to state [2]
transition on ClassDecl to state [7]
transition on Type to state [6]
transition on DeclList to state [11]
```

(i) Write down the set of items for goto(state 10, Decl) (state 9).

(5 marks)

(ii) Write down the set of items for goto(state 10, Type) (state 6). Make sure you take the closure.

(8 marks)

Question 3 20 marks

Write a grammar for the **general** structure of a state description.

Print your login name _____